Generalization by Default: A Finite Universal Simplicity Prior

Matthias Dellago

July 2025

Why do learning systems generalize? The Solomonoff prior is our current best theoretical answer; it formalizes Occam's razor into an exponential bias towards simpler hypotheses. Crucially, its proof requires that both the data source and the learner be Universal Turing Machines (UTMs), idealized computers with infinite memory and time. And yet, all empirical observations of generalization—biological neural networks and machine learning algorithms—occur exclusively in physical, finite systems.

Dispensing with the unphysical requirements of UTMs, we model finite learners with transformation semigroups, an algebraic framework that immediately applies to neural networks and all finite computational systems. Within this framework, we prove finite analogues of the Solomonoff prior and Kolmogorov invariance:

1. An exponential simplicity prior on ideals (absorbing sets, corresponding to domain partitions), and
2. Invariance of this prior across different generator sets of the same semigroup, up to multiplicative slack.

Intuitively: Given a set of computational primitives, certain distinctions in the input domain are simpler to express and therefore exponentially more probable to be computed. Furthermore, different computational primitives that can emulate each other exhibit an equivalent bias toward simpler computations, within bounds. This implies that learners capable of emulating their target system inherently acquire the appropriate simplicity prior.

1 Introduction

Learning systems generalize from finite data, yet our primary theoretical framework for understanding this phenomenon — the Solomonoff Prior — relies critically on the idealized assumption of infinite computational resources (Universal Turing Machines). This creates a fundamental disconnect with the empirical reality of generalization in physical, resource-bounded systems like biological neural networks or practical machine learning algorithms. We resolve this disconnect by modelling finite learners through the algebraic framework of transformation semigroups. Within this framework, we establish finite analogues of the Solomonoff prior's core elements: an exponential simplicity prior over absorbing structures (ideals) and invariance under changes of computational primitives (generator sets).

We will first introduce the problem of induction and the Solomonoff prior in appropriate detail (Sec. 2). We will then review our tools, semigroups in the preliminaries, Sec. 3, before showing our main results in Sec. 4. Afterwards we discuss the implications for deep learning and the connections to other fields, including circuit complexity (Sec. 5) before turning to future extensions and refinements (Sec. 6. In the appendix, we attach a running example to ease the unfamiliar reader into transformation semigroups (Sec. B), and some references to related work (Sec. A).

2 Background

2.1 Inductive Inference and The Necessity of a Non-Uniform Prior

Inductive inference, the process of generalizing from past observations to predict future events, underpins learning and scientific discovery. The crux, as articulated by Hume, is that any finite sequence of observations is logically consistent with an infinite number of possible continuations and underlying generative processes [12]. Hume called this 'the problem of induction'. Nowadays it most commonly appears as the problem of 'generalisation' in machine learning. This underdetermination implies that predicting the future from the past inherently ill-posed.

Yet, empirical reality obviously exhibits significant structure and predictability; Everyday cognition and scientific progress show successful generalization from finite data is not only possible, but common. Thus, effective inductive systems (biological or artificial) cannot treat all logically possible futures or explanations as equally probable. Instead, they must choose non-uniform priors, weighting certain hypotheses higher than others a priori.

Historically, the best solution to this was Occam's Razor: a bias for preferring the 'simpler' solution whenever possible [26], but formally defining 'simplicity' remained a persistent challenge for centuries. [1] The breakthrough came

[1] Anthropic reasoning might explain why observers must inhabit a learnable universe, but

in the 1960s when Solomonoff, Kolmogorov, and Chaitin converged on an objective notion of complexity based on universal Turing machines [29, 16, 2, 31]. This quantity, known as Kolmogorov complexity K, quantifies the simplicity of an object (like a data sequence or hypothesis) by the length of the shortest computer program required to generate it.

Solomonoff's work culminated in a universal simplicity prior, offering a formal instantiation of Occam's Razor [29, 30]. The following sections explore Solomonoff's prior in more detail: its foundational assumptions (Section 2.2), the core theorems and their proof mechanisms (Section 2.4), and its limitations (Section 2.6). This overview is not a comprehensive treatment of Algorithmic Probability. Instead, it introduces foundational concepts and mechanisms relevant to the methods subsequently developed in this work. For exhaustive treatments, the reader is directed to established literature, e.g., [20, 13].

2.2 Foundational Assumptions of the Solomonoff Prior

The Solomonoff Prior answers the inductive inference question: *Assuming our observations are generated by an unknown computable process, what future observations should we expect?* To derive it, we must first specify two assumptions:

2.2.1 Assumption 1: Computable Processes

A *computable process* is any procedure that can be implemented as a program on a Universal Turing Machine (UTM) [31]. Formally, a UTM takes a binary program p as input and simulates the execution of p, outputting a (potentially infinite) sequence of observations. Crucially, by the Church-Turing thesis [3], any computable function can be represented this way. Thus, the hypothesis space for Solomonoff induction consists of all possible UTM programs. While the choice of UTM affects program lengths, the Kolmogorov invariance theorem (Sec. 2.2) ensures this choice is asymptotically negligible.

2.2.2 Assumption 2: The Principle of Epicurus

The second assumption is the *Principle of Epicurus* [5]: all hypotheses (in our case, programs) consistent with the observed data are retained until contradicted. This principle avoids premature elimination of potential explanations, however improbable. Consequently, the prior assigns non-zero probability to every computable hypothesis.

2.3 Kolmogorov Complexity K

To construct a simplicity prior, we first require a rigorous measure of "simplicity." Intuitively, the defined complexity of an object (e.g., a hypothesis, function, or dataset) will correspond to the shortest program needed to describe it.

not how learning occurs or priors are formed.

Definition 2.1 (Kolmogorov Complexity)**.** Let U be a universal Turing machine. The *Kolmogorov complexity* $K(x)$ of a finite binary string x is the length of the shortest program p for U that outputs x and halts:

$$K(x) = \min \{\texttt{len}(p) \mid U(p) = x\},$$

where $\texttt{len}(p)$ denotes the length of program p in bits.

By convention, $K(x)$ is defined on some canonical choice of Turing machine. The invariance theorem (Sec. 2.2) ensures this definition is machine-independent up to an additive constant.

Extensions and Subtleties. While Definition 2.1 applies to finite strings, complexity can be extended to other objects (e.g., functions, hypotheses) by encoding them as strings. Since handling this requires some care, but the technicalities are ultimately not relevant for our purpose, we defer to 'An introduction to universal artificial intelligence' for a rigorous treatment [13].

2.4 Central Results of Algorithmic Probability

Having established the foundational assumptions and the formal notion of complexity, we now present the central results of algorithmic information theory, both for context and because we will later use similar techniques in our proofs.

2.4.1 Exponential Simplicity Prior

Theorem 2.1 (Solomonoff Prior)**.** Let U be a universal Turing machine and h be a hypothesis. The universal prior probability of h is bounded by:

$$P_U(h) \geq 2^{-K_U(h)-O(1)},$$

where $K_U(h)$ denotes the Kolmogorov complexity of h with respect to U.

This result formalizes Occam's Razor: the prior probability of a hypothesis decreases exponentially with its Kolmogorov complexity [29, 30].

2.4.2 Machine Independence

A natural concern arises: the prior appears dependent on the choice of UTM U. Since we assume only that the generating process is computable, without specifying the computational model, this dependence might seem problematic for truly *universal* inference. The following theorem resolves this issue:

Theorem 2.2 (Kolmogorov Invariance Theorem)**.** For any two universal Turing machines U and V, there exists a constant $c(U, V)$ such that for all finite strings h:

$$|K_U(h) - K_V(h)| \leq c(U, V).$$

The constant $c(U, V)$ depends only on the machines U and V, not on h.

Consequently, Kolmogorov complexity is machine-independent up to an additive constant [16]. For the Solomonoff prior, this translates to machine independence up to a multiplicative constant: $P_U(h)$ and $P_V(h)$ differ by at most a factor of $2^{c(U,V)}$. This invariance justifies the universality of algorithmic probability for inductive inference [20].

2.5 Proof Ideas

To understand why these theorems hold, we sketch the central ideas underlying their proofs. Analysing these proof mechanisms will enable us to apply similar techniques to alternative models of computation in subsequent sections. For rigorous treatments, we refer to [20, 13].

2.5.1 Kolmogorov Invariance: Emulation Begets Invariance

Proof Sketch. The key to this proof is that a universal Turing machine can emulate any other Turing machine, including another UTM. Given two UTMs U and V, we can find a finite prefix p_{UV} that programs V to behave like U on subsequent input. This prefix p_{UV} effectively *compiles* programs from U to V:

$$U(x) = V(p_{UV} + x), \quad \forall \text{ programs } x,$$

where $+$ denotes concatenation. Thus, the complexity of any string h in U is upper bounded by the emulation overhead plus the complexity in V:

$$K_U(h) \leq \texttt{len}(p_{UV}) + K_V(h).$$

By symmetry, a similar bound holds in the reverse direction with emulation prefix p_{VU}. Therefore, we can define a constant $c(U, V) = \max\{\texttt{len}(p_{UV}), \texttt{len}(p_{VU})\}$ that bounds their difference.

Takeaway: *Emulation* between UTMs ensures complexity invariance up to constant overhead.

2.5.2 Solomonoff Prior: Absorption Begets Exponential Decay

Proof Sketch. The Padding Argument

Consider the shortest program generating h on a UTM, having length $K(h)$. For programs of fixed length $L > K(h)$, we count how many output h.

First, append a halting instruction (e.g. `exit`) to the shortest program, requiring $K(\texttt{exit})$ additional bits. Any bits appended after this instruction are *absorbed*: they do not affect the output. Thus, the remaining $L - K(h) - K(\texttt{exit})$ bits are irrelevant padding. Since we can set each padding bit either way this yields $2^{L-K(h)-K(\texttt{exit})}$ distinct programs of length L that output h.

The probability that a random program of length L outputs h is therefore at least:

$$\frac{2^{L-K(h)-K(\texttt{exit})}}{2^L} = 2^{-K(h)-O(1)}.$$

This provides only a lower bound[2], as completely unrelated programs on other branches of the binary tree may also generate h:

$$P_U(h) \geq 2^{-K_U(h)-O(1)}.$$

Takeaway: *Absorption* of excess program bits yields exponentially many equivalent programs, offsetting the exponential growth of the program space.

2.6 Shortcomings of the Solomonoff Prior

The Solomonoff prior has several fundamental problems:

- **Physical implausibility**: UTMs require unbounded memory and time, contradicting physical reality. The framework cannot directly explain finite-resource learning in neural networks or biological systems.
- **Uncomputability**: Because of the infinite resource requirements$K$ is uncomputable [16, 2]: determining the shortest program for any string requires solving the halting problem. Thus the Solomonoff prior is uncomputable even in theory.
- **Machine dependence**: While invariance holds up to a constant, this constant $c(U, V)$ can be arbitrarily large for pathological UTMs, making the prior unusable in practice [13].
- **Explanatory disconnect**: Why should string concatenation and prefix codes reveal deep truths about induction? The mechanism seems divorced from actual learning processes.

These limitations necessitate a new approach. The following section introduces transformations, and their algebraic structure—semigroups—which will form the foundation for a physically plausible, finite simplicity prior.

3 Preliminaries

Now that we have isolated and understood the mechanisms behind the invariance theorem and the simplicity prior we propose another more physical system and show analogous results.

This section lays the groundwork by introducing the core mathematical objects:

- Transformation semigroups
- Multiplicity: the number of ways an element can be decomposed into some generating elements
- Probability: normalized multiplicity
- Ideals: absorbing sets of a semigroup

[2]An upper bound follows from the Kraft-McMillan inequality [17, 22].

3.1 Deep Learning and Physics as Transformations on States

Deep learning and physical systems share a fundamental property: they change state through sequences of transformations. In physics these primitive transformations are called propagators, time evolution operators or state transition operators. In deep learning the set of primitive transformations corresponds all distinct parametrisations of a layer.

Assumption 3.1. We consider a system with a state space X and a set of transformations G called *generators*. At each time step, a generator $g \in G$ transforms the state, where each $g : X \to X$.

In a constant width neural network e.g., the state space X corresponds to the activation space, and the generators G represent the different parametrizations of a single layer.

3.2 Tangent: Relation to Automata

Incidentally this construction is very similar to discrete finite automata [15, 27], if we take X as the states, G as the alphabet, and the transition function is given by applying each g to the current state.[3]

Notably absent are the **start state** and the **set of final states**. We also do not require finite states or alphabet. These are crucial for *language recognition* (determining if a specific run on an input string is accepting), but they do not change the underlying set of possible transformations or the structure of the transitions : Our interest lies only in the properties of these transformations and how they compose - their algebraic structure. The central object in our analysis will therefore not be the automata, but the semigroup associated with each.

Of course, the results obtained from studying the semigroup nevertheless directly apply to understanding the dynamics inherent in the corresponding automaton. The algebraic study of automata, via their semigroups or monoids is well established, the most notable result of this duality being Krohn-Rhodes Theory [18].

3.3 Transformation Semigroups

We will now introduce the semigroup generated by our generator transformations. For the reader inclined to learn by example, we have supplied an accompanying running example in Appendix B, which illustrates the key concepts.

When considering the evolution of our system over multiple time steps, transformations are naturally chained together. We can express this via function composition $\circ$:

$$(g_2 \circ g_1)(x) = g_2(g_1(x))$$

Note the right to left execution order.

[3] Readers familiar with automata theory will also recognise this as a *semiautomaton* [4]. In the operational semantics literature this is also often called a *labelled transition system* [14].

Definition 3.1 (Transformation Semigroup)**.** Let the semigroup T be the set of all possible transformations that can be *generated* by composing elements of G:

$$T = \{g_n \circ \cdots \circ g_2 \circ g_1 \mid g_i \in G, n \in \mathbb{N}\}$$

In neural networks, T is the set of all functions that can be expressed by composition of finitely many layers. In physics it is the set of transformations a system can undergo if G is the set of possible propagators.

Since function composition is necessarily associative, and T is by definition closed under composition these transformations form a *semigroup* under composition $(T, \circ)$ [4]. Semigroups are a well-known algebraic structure, called so because they generalise Groups, which are closed, associative, have an inverse and an identity. Semigroups relax these requirements and only retain the first two axioms.

3.4 Paths, Multiplicity and Probability

Having defined the set of all possible transformations T, we can study how each transformation arises from the compositions of generators. For any transformation $t \in T$, there necessarily exists at least one sequence of generators $(g_1, g_2, ..., g_L)$, where $g_i \in G$ for $i = 1, \ldots, L$, such that:

$$t(x) = (g_L \circ ... \circ g_2 \circ g_1)(x) \quad \forall x \in X$$

We call such a sequence a *path* of length L generating t. In deep learning L corresponds to the number of layers.

In general, there can be many paths that generate the same transformation. So, given a composition of L generators, how many distinct paths generate a given transformation t?

Definition 3.2 (Multiplicity)**.** The multiplicity $\Omega_L(t)$ of a transformation t is the number of distinct paths of length L that generate t.

$$\Omega_L(t) = |\{(g_L, ..., g_2, g_1) \mid g_i \in G, t = g_L \circ ... \circ g_2 \circ g_1\}|$$

Multiplicity is also sometimes called *degeneracy*.

In NNs this corresponds to: "How many different ways can I compose function t in L layers?"

Since there are $|G|$ choices for at each position in a path of length L, the total number of possible paths grows as $|G|^L$. To make meaningful comparisons across different systems and path lengths, we normalize the multiplicity by this total:

Definition 3.3 (Probability)**.** The probability $P_L(t)$ of a transformation t at length L is its multiplicity normalized by the total number of possible paths:

$$P_L(t) = \frac{\Omega_L(t)}{|G|^L}$$

This definition is analogous to algorithmic probability theory, where each program/code is viewed as equally likely. Here, under the assumption of uniformly sampling generators, $P_L(t)$ gives the probability of randomly generating transformation t through a composition of L generators.

Equivalently we can also think of this probability as the result of an L step random walk on the transition semigroup [1, 9].

In deep learning this corresponds to the probability of drawing function t from a uniform prior over parameter space of L layers.

An important property of any transformation is how concisely it can be expressed in terms of our generators. This leads us to define complexity in a way analogous to algorithmic information theory:

Definition 3.4 (Composition Complexity)**.** The composition complexity $C(t)$ of a transformation $t \in T$ is the minimum composition length , i.e. the smallest number number of generators required to compose it:

$$C(t) = \min\{L \mid \exists(g_1, g_2, \ldots, g_L) \in G^L \text{ such that } t = g_L \circ \cdots \circ g_1\}$$

Or more succinctly: the minimum length L for which there exists at least one way to generate t:

$$C(t) = \min\{L \mid \Omega_L(t) > 0\}$$

In deep learning, this corresponds to the minimum depth of a neural network required to implement a given function.

Importantly, complexity depends on our choice of generators, analogously to how in AIT complexity (program length) depends on the choice of universal Turing machine. We will prove an analogue to the Invariance theorem in Lemma 4.1

In summary, we have defined a framework to study the transformations of systems that is very general and notably *not* Turing complete. This is precisely what we set out to do, since physical systems nor neural networks are Turing complete, due to limited space (state space size $|X|$) and time (composition length L).

3.5 Zeros and Ideals

Via the padding argument proof sketch (2.5.2), we understood that the underlying mechanism of the Solomonoff prior was *absorption*. We will now apply this idea to semigroups.

We are looking for an absorbing element in a semigroup; whatever transformation you apply afterwards it does not change the implemented function. In algebraic terms this is called a *left zero*:

Definition 3.5 (Left Zero)**.** An element $\overleftarrow{0}$ is called a *left zero* iff

$$\forall t \in T : t \circ \overleftarrow{0} = \overleftarrow{0}.$$

The arrow on top indicates the direction of action.

After such a left zero adding more elements does not change the result, e.g.:

$$d \circ c \circ \overleftarrow{0} \circ b \circ a = \overleftarrow{0} \circ b \circ a$$

(Reminder: the left most transformation is applied first.)

In a NN such a forward zero would correspond to an 'early exit' or somehow 'locking in' the activations so that subsequent layers can no longer affect them.

Zero elements are particularly interesting from a multiplicity perspective: Any transformation that ends in a forward zero (e.g. $\overleftarrow{0} \circ t$), will have exponentially increasing multiplicity beyond $C(\overleftarrow{0} \circ t)$, and constant probability. All suffixes will simply be absorbed into the left zero.

3.6 Ideals, Domain Partitioning and Information Loss

Such absorbing elements are rare in physical systems, but we can generalise the notion of a zero, an absorbing *element*, to an absorbing *set*. This is called an *Ideal*:

Definition 3.6 (Left Ideal)**.** A subset $\overleftarrow{I} \subset T$ is called a *left ideal* iff

$$t \circ \overleftarrow{i} \in \overleftarrow{I} \quad : \forall t \in T \text{ and } \forall \overleftarrow{i} \in \overleftarrow{I}$$

Surprisingly left ideals have a very intuitive interpretation, that is particularly relevant to machine learning: in transformation semigroups left ideals correspond to partitionings of the domain. Once a transformation maps a set of values to the same output, applying any subsequent transformation cannot separate them again. The resulting composition must remain in the ideal, thus it becomes an *absorbing set*. We can think of this as irrecoverable loss of information!

At this point the relevance to (machine) learning might become more apparent: If we think of e.g. classification tasks, partitioning the input space into decision regions is the challenge. If we don't care about the specific labels each is assigned (like ideals) to we can think of this as clustering.

All these concepts are illustrated in Fig. 1 , a simple transformation semigroup, with generators and ideals. For a more detailed explanation please refer to the running example in Appendix B.

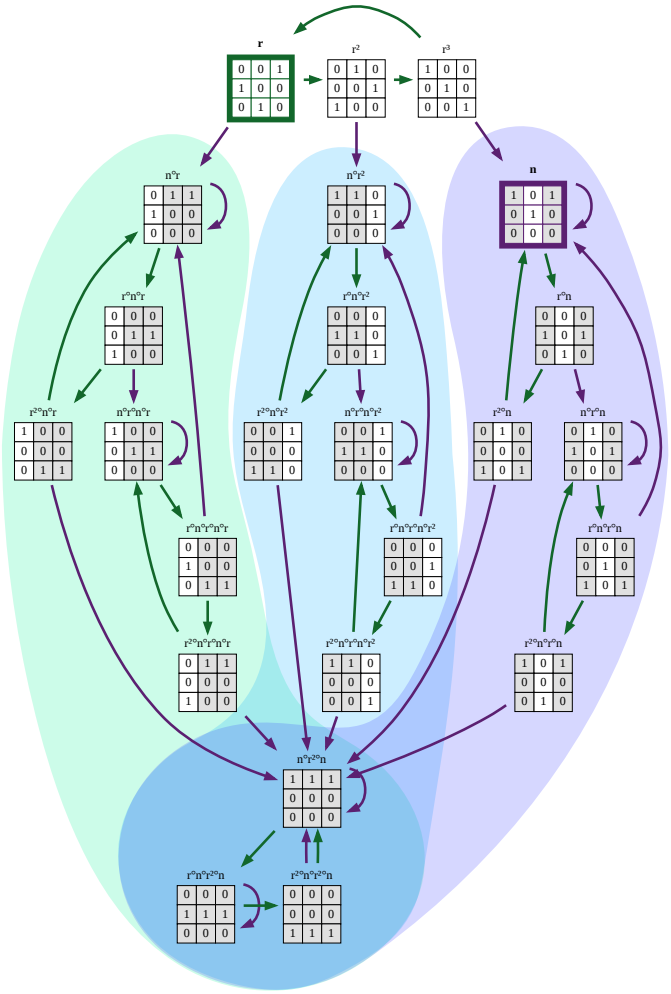

Figure 1: This is the semigroup generated by transformations n and r. Each transformation is represented by a matrix, acting on the state space $X = \{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$. Purple arrows represent application of n, green arrows of r. The ideals are represented as coloured regions: We notice that arrows never point *out* of a coloured region only in. We also observe that each of the regions corresponds to reduction in the range: identical columns (highlighted in grey) mean that two input states are mapped to the same output. The intersection of all ideals form the minimal ideal: the erasing of all distinction — all states are mapped to a single output.

3.7 Extending Definitions to Sets

Since our theory will be one of absorbing sets rather than elements we must extending our previous definitions of probability and multiplicity from elements to sets.

Definition 3.7 (Multiplicity of Sets)**.** The multiplicity $\Omega_L(S)$ of a of a Subset $S \subset T$ is the number of distinct paths of length L that end in S,

$$\Omega_L(S) = \sum_{s \in S} \Omega_L(s)$$

Definition 3.8 (Probability of Sets)**.** Again, we just normalize by the total multiplicity:

$$P_L(S) = \frac{\Omega_L(S)}{\Omega_L(T)} = \frac{\Omega_L(S)}{|G|^L}$$

Definition 3.9 (Complexity of Sets)**.** The shortest path that leads to an element of the set, the minimum complexity of all element

$$C(S) := \min_{s \in S} C(s) = \min\{L \mid \Omega_L(S) > 0\}$$

3.8 Parallels to AIT

For convenience we have listed some of the analogies between our theory and AIT in Table 1.

Table 1: Analogues to AIT in our framework.

K Kolmogorov Complexity	C Composition length
Universal Turing Machine	G Generator set
UTMs U and V	Generator sets G and H
Set of all possible hypotheses/codes $\mathcal{H}$	Set of all possible transformations T
(self-)terminating code	Absorbing set (left ideal) I
Emulator p_{UV}	Emulation overhead $O_{G \to H}$

4 Results

Having established the algebraic framework of transformation semigroups and their absorbing sets (ideals) in the previous section, we now present our main theoretical contributions:

- **Theorem 4.1** establishes an exponential simplicity prior for semigroup ideals: $P(I) \geq |G|^{-C(I)}$. Unlike Solomonoff's prior which assigns probabilities to individual hypotheses, our result pertains to ideals, equivalence classes of transformations.

- **Theorem 4.2** proves generator invariance analogous to Kolmogorov invariance, demonstrating that complexity ratios remain bounded under changes of generator sets: $\frac{1}{O_{H\to G}} \leq \frac{C_H(t)}{C_G(t)} \leq O_{G\to H}$.

Together, these results establish a simplicity prior for transformation semigroups that parallels Solomonoff's framework while remaining applicable to physically realizable, resource-bounded systems such as neural networks and finite automata.

4.1 Probability of Ideals: A Generalised Simplicity Prior

The absorption property of ideals provides the key mechanism for establishing an exponential decay in probability, analogous to the padding argument in Solomonoff's framework. We now formalize this intuition.

Theorem 4.1 (Ideal Simplicity Prior)**.** Let $I \subseteq T$ be a left ideal of a transformation semigroup with generator set G. For any path length $L \geq C(I)$, the probability mass of I satisfies:

$$P_L(I) \geq |G|^{-C(I)}$$

where $C(I) = \min_{i\in I} C(i)$ is the complexity of the ideal.

Proof. Let $i^* \in I$ be an element of minimal complexity, so $C(I) = C(i^*)$ by definition. Any extension of i^* will remain in I because I is a left ideal: for any $t \in T$ and $i \in I$, we have $t \circ i \in I$.

Consider paths of length $L \geq C(I)$. Any path that generates i^* in $C(I)$ steps can be extended with arbitrary generators for the remaining $L - C(I)$ steps, yielding $|G|^{L-C(I)}$ distinct paths that end in I.

Thus the multiplicity of I satisfies:

$$\Omega_L(I) \geq |G|^{L-C(I)}$$

The probability follows by normalization:

$$P_L(I) = \frac{\Omega_L(I)}{|G|^L} \geq \frac{|G|^{L-C(I)}}{|G|^L} = |G|^{-C(I)}$$

Since this bound is independent of L (for all $L \geq C(I)$), we obtain the stated result. □

Remark The theorem establishes that simple ideals—those reachable by short compositions—dominate the hypothesis space exponentially. As discussed in Section 3.6, ideals represent irreversible domain partitionings where distinct inputs merge and cannot be separated by subsequent transformations. This provides a natural explanation for why we commonly observe 'simple' decision boundaries and information bottlenecks, and why learning systems are biased towards these.

4.2 Invariance and Emulation

As established in Section 2.2, the machine independence of Kolmogorov complexity—achieved through emulation between UTMs—is crucial for the universality of Solomonoff's prior. We now demonstrate an analogous invariance for transformation semigroups: complexity remains stable (up to multiplicative factors) under changes of generator sets, provided they can mutually emulate each other. This ensures our simplicity prior is not an artifact of a particular choice of generators, but reflects intrinsic properties of the transformation semigroup itself.

Lemma 4.1 (Emulation Overhead)**.** Let G and H be two sets of transformations where G generates semigroup T. If each $g \in G$ can be expressed by a composition of at most $O_{H\to G}$ elements from H, then for any transformation $t \in T$:

$$C_G(t) \leq O_{H\to G} \cdot C_H(t)$$

Proof. Let $t = h_k \circ \cdots \circ h_1$ be a minimal representation of t using $k = C_H(t)$ generators from H. Since each h_i can be expressed using at most $O_{H\to G}$ generators from G, substituting these expressions yields a representation of t using at most $k \cdot O_{H\to G} = C_H(t) \cdot O_{H\to G}$ generators from G. The minimal length $C_G(t)$ must be at most this construction. □

Theorem 4.2 (Complexity Invariance)**.** Let G and H be generator sets for the same semigroup T with mutual emulation overheads:

$$O_{H\to G} = \max_{h\in H} C_G(h) \quad \text{and} \quad O_{G\to H} = \max_{g\in G} C_H(g)$$

Then for any transformation $t \in T$ with $C_G(t) > 0$ and $C_H(t) > 0$:

$$\frac{1}{O_{H\to G}} \leq \frac{C_H(t)}{C_G(t)} \leq O_{G\to H}$$

Proof. Applying Lemma 4.1 directly gives $C_G(t) \leq O_{H\to G} \cdot C_H(t)$. Since complexities and overheads are positive, rearranging yields: $C_H(t)/C_G(t) \geq 1/O_{H\to G}$.

Applying Lemma 4.1 with roles reversed gives $C_H(t) \leq O_{G\to H} \cdot C_G(t)$, yielding: $C_H(t)/C_G(t) \leq O_{G\to H}$. □

Consequences This invariance extends immediately to ideal complexities, ensuring our simplicity prior (Theorem 4.1) remains valid across different generator choices. The relative complexity ordering of transformations—and hence their prior probabilities—is preserved up to the emulation overhead factors.

Having established a generalized simplicity prior based on ideal complexity (Theorem 4.1) and an invariance theorem for transformation complexity (Theorem 4.2), we now turn to discuss the broader implications of these findings and their potential applications (Section 5).

5 Discussion

5.1 Relation to Deep Learning

To illustrate how these results apply to deep learning, consider a biological neural network (BNN) classifying images into $\{\text{cat}, \text{dog}, \text{other}\}$, implementing the function f through a series of physical transformations on the neurons state. Suppose we construct an artificial neural network (ANN) where each BNN transformation can be expressed by compositions of at most 3 ANN layers (emulation overhead $O = 3$). Our framework yields two key guarantees:

1. **Complexity Transfer** (Lemma 4.1):

$$C_{\text{ANN}}(f) \leq 3 \cdot C_{\text{BNN}}(f)$$

   The complexity of f in the ANN is bounded by the complexity in the BNN.

2. **Ideal Probability Bound** (Theorem 4.1): For ANNs of sufficient depth ($L \geq 3 \cdot C_{\text{BNN}}(f)$), the probability of the smallest ideal containing f, called I_f, is bounded from below:

$$P_L(I_f) \geq |G|^{-3 \cdot C_{\text{BNN}}(f)},$$

   where $|G|$ is the number of distinct ANN layer parametrizations. I_f contains f and all functions that collapse *at least* the same input distinctions as f (i.e., induce partitions equal to or coarser than f's). In the usual terminology of deep neural networks we would not interpret this as a probability but as a lower bound on the relative volume this ideal must occupy in parameter space.

In general, this demonstrates that to the ability to emulate the computations of the physical target system necessarily implies that simple transformation are associated with exponentially large measures of the models hypothesis space. When this multiplicity concentrates locally, it explains observed degeneracy phenomena like flat minima and singularities in the loss landscape.

5.2 Connection to Established Definitions of Complexity

5.2.1 Circuit Complexity

Circuit-depth complexity[36] is straightforwardly a special case of our notion of compositional complexity: all parallel combinations of the allowed boolean gates form the generator set of the relevant semigroup. Furthermore, gate-count complexity[28] as depth complexity weighted by the number of gates the generator. Theorems from this field may also apply to the more general setting of semigroups.

5.2.2 VC Complexity

Both frameworks analyze domain partitioning, but with complementary perspectives: VC dimension measures a hypothesis class's capacity to shatter arbitrary subsets of the domain (preserved distinguishability) [33], while ideals characterize irreversible merging of domain elements under transformation semigroups (information loss). This reveals a fundamental duality: VC dimension bounds the maximum number of distinguishable configurations, whereas ideals emerge when transformations collapse distinctions. The algebraic structure of ideals may thus provide a mechanistic foundation for the capacity limitations observed in VC theory.

6 Future Work

6.1 An Upper Bound

The most notably theory lacks an upper bound on the prior. In the UTM case this is accomplished by the Kraft-McMillan inequality [17, 22]. In the case of ideals this is not possible, because generator compositions do not uniquely encode ideals: Because ideals are not disjoint an element may be part of multiple ideals.

The Green's Relations [6] provide a more subtle instrument for analysing ideal structure, and it may be possible to formulate a prior over Green's *R-classes* with an upper bound with them, because they are disjoint.

This is motivated from the fact that destroying more information necessitates a reduction of the transformation range. Therefore, conditioning on the range would remove all transformation in the ideal that further merge the domain partitions. This would leave a prior over R-classes, transformations equivalent up to relabelling of outputs,.

6.2 The Minimal Ideal and Landauer's Limit

Additionally, we would also expect all the probability mass to flow to the minimal ideal in the limit of long compositions, e.g. mapping the whole domain to a single output, destroying all information. But this is not what we empirically observe. What 'pressure' drives away from collapse? We can argue via Landauer's limit that it takes a minimum of energy to destroy information [19]. For a system with a limited amount of energy, or in thermal equilibrium, this could result in a non-trivial distribution over R-classes. (As opposed to total absorption into the minimal ideal.)

6.3 Applying Krohn-Rhodes

Besides Green's relations one could also bring many other algebraic tools to bear. Krohn-Rhodes Theory [18] could be used to relate this work to automata and to understand how the prior of decomposable semigroups behaves. This is

a natural path to investigate how this prior behaves under coarse-graining or approximation, and how modularity contributes to degeneracy.

7 Conclusion

This work establishes finite analogues of the Solomonoff Prior and Kolmogorov Invariance through the algebraic framework of transformation semigroups. By modelling physical and neural systems as compositions of state transformations, we demonstrate:

1. an exponential simplicity prior over ideals, and
2. complexity invariance across generator sets.

These results extend the ideas of algorithmic probability to resource-bounded systems.

Three constraints distinguish our results from classical AIT: priors operate on ideals (equivalence classes of information loss) rather than individual hypotheses; only lower probability bounds exist due to non-disjoint ideals; invariance holds up to multiplicative rather than additive constants. We point to algebraic and thermodynamic approaches to remedy these limitations. Unlike AIT, our complexity, and therefore the prior, as well as the emulation constants are computable.

For deep learning, this provides a new lens to analyse degeneracy in hypothesis space: Equivalence classes of transformations that discard specific input distinctions (ideals) dominate the parameter space volume exponentially. Additionally, it suggests that the ability of these large networks to emulate the computations that generated their training data is absolutely essential and inherently induces the correct prior and thus their ability to generalise.

8 Acknowledgements

Thank you to The Long Term Future Fund for funding this research. I am very grateful for the support of Alexander Gietelink Oldenziel, Artūrs Bērziņš, Bastian Bartel, Can Rager, Clément Dumas, David Quarel, Effectief Altruïsme NL, Erik Bekkers, Florian Scheidl, Gerald Gutenbrunner, Jaffar Hasnain, Jana Abuasbeh, Johannes Brandstetter, Jonathon Liu, Justus Piater, Korbinian Poeppel, Lucius Bushnaq, Marcello Barylli, Marcus Hutter, Max Hennick, Salvatore Romano, Sepp Hochreiter, Xaver Henneberger, and my mother, father and sister.

A Related Work

Our approach can be summarized as tackling the biases and degeneracies in the parameter function map, using proof techniques from AIT, in the framework of transformation semigroups. In the following we will briefly review the most relevant results from these three fields.

A.1 Parameter-Function Map

The connection between degeneracy and generalisation was first described in 'Flat Minima' by Hochreiter and Schmidhuber [8].

The theoretical treatment of singular models dates back to Watanabe's Singular Learning Theory [34, 35], which has recently experienced a resurgence and is being applied to modern deep learning [37, 10]. Murfet and Troiani described a possible connection to AIT [25].

Recent work indicates that this map is multi-fractal [7, 21].

A.2 Simplicity Biases in Deep Learning

Previous work has attempted to directly connect algorithmic information theory to deep learning. Valle-Pérez et al. argued that the parameter-function map of DNNs is exponentially biased towards simple functions, applying probability-complexity bounds from AIT to explain generalization [32].

Mingard et al. demonstrated that DNNs possess an inbuilt Occam's razor, showing that the prior over functions induced by network architecture favors Kolmogorov simple functions [23].

For Boolean functions specifically, Mingard et al. proved that even single-layer perceptrons exhibit strong a priori bias towards low entropy functions, with this bias becoming monotonically stronger upon adding ReLU layers [24].

This line of work relies on Kolmogorov Complexity and thus UTMs, which is a computational model strictly more powerful than NNs.

A.3 Transformation Semigroups

The fundamental connections between finite automata and semigroups was established by Krohn-Rhodes decomposition theorem [18], providing a foundation for applying algebraic methods to computational systems.

Hryniowski and Wong applied applied finite transformation semigroup theory to analyze convolutional neural networks [11]. In contrast to their work we map models to semigroup elements.

Earlier work studied random walks on finite semigroups, examining the structure of transition matrices and their relationship to the algebraic properties of the underlying semigroup [1].

Högnäs and Mukherjea provided a comprehensive treatment of probability measures on semigroups, establishing connections between probabilistic proper-

ties of random walks and the algebraic structure of their supporting semigroups [9].

B Running Example

Here we will demonstrate all the concepts and definitions of Section 3 using a 'tipping bucket' system with three possible states: empty, half-full, and full. Let's represent each state as a one-hot vector: empty $(1,0,0)^T$, half-full $(0,1,0)^T$ and full $(0,0,1)^T$. So the state space is $X = \{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$.

The system evolves through two generators G:

1. Rain r: Adds half a bucket of water.
   - Empty to half-full: $(1,0,0)^T \mapsto (0,1,0)^T$
   - Half-full to full: $(0,1,0)^T \mapsto (0,0,1)^T$
   - Full bucket tips over: $(0,0,1)^T \mapsto (1,0,0)^T$
2. Nudge n: If a bucket is full and unstable, a nudge will tip it over.
   - Full bucket tips over: $(0,0,1)^T \mapsto (1,0,0)^T$
   - Other states remain: $(1,0,0)^T \mapsto (1,0,0)^T$, $(0,1,0)^T \mapsto (0,1,0)^T$

Because we chose one-hot vectors to represent the states, we can now conveniently represent the operations as matrices:

$$r = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad n = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

By convention we apply the matrices from the left. We can easily see what state each input is mapped to, by remembering that the column of a matrix are the images of each basis vector.

Composing our state transition functions is equivalent to matrix multiplication. For example, composing nudge n with rain r yields:

$$n \circ r = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

And composing rain r with itself three times yields:

$$r \circ r \circ r = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \mathbb{1}$$

In fact, by chaining together r and n generators, we can generate 24 different transformations on this three-state system.

The composition $n \circ r \circ n$ can be visualized as a path:

$$n \xrightarrow{r} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{n} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

In fact we can display all possible paths that can be taken through the semigroup as a directed graph, Fig. 2.

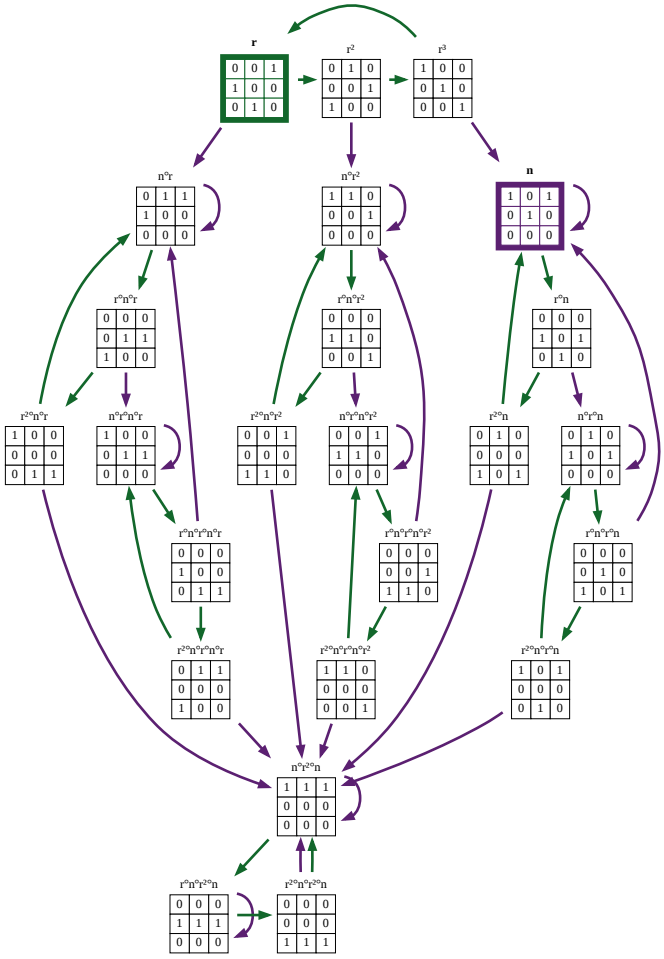

Figure 2: Bucket semigroup directed graph. Purple arrows represent application of n, green arrows of r.

Consider the transformation

$$n = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The images of the three states are easy read off: they correspond to the columns. We can see that n is not a one-to-one, mapping both $(0,0,1)^T$ and $(1,0,0)^T$ to $(1,0,0)^T$. If we apply other transformations afterwards, these two domain elements can be mapped to other values, but never separated again. We can think of this as irrecoverable loss of information!

In other words; n does not absorbs into itself, but instead into the set of all transformations that map $(0,0,1)^T$ and $(1,0,0)^T$ to the same value. These absorbing sets are called ideals. In Fig. 3, these are represented by coloured regions.

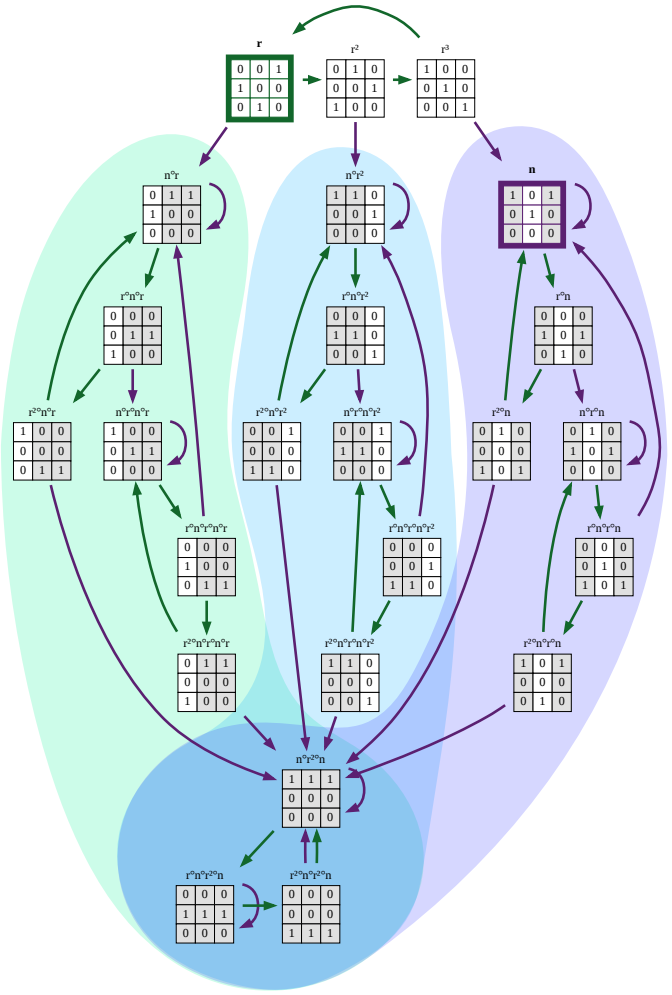

Figure 3: Bucket semigroup ideals. The ideals are represented as coloured regions: We notice that arrows never point *out* of a coloured region only in. We also observe that each of the regions corresponds to reduction in the range: identical columns (highlighted in grey) mean that two input states are mapped to the same output. The intersection of all ideals form the minimal ideal: the erasing of all distinction — all states are mapped to a single output.

References

[1] George R. Barnes, Patricia B. Cerrito, and Inessa Levi. Random walks on finite semigroups. *Journal of Applied Probability*, 35(4):824–832, December 1998.

[2] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, October 1966.

[3] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345, 1936.

[4] A. Clifford and G. Preston. *The Algebraic Theory of Semigroups, Volume I.* American Mathematical Society, December 1961.

[5] Epicurus. Letter to pythocles, circa 3rd Century BCE. Preserved in Diogenes Laertius, Lives of Eminent Philosophers, Book X.

[6] J. A. Green. On the structure of semigroups. *The Annals of Mathematics*, 54(1):163, July 1951.

[7] Max Hennick and Stijn De Baerdemacker. Almost bayesian: The fractal dynamics of stochastic gradient descent, 2025.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, January 1997.

[9] Göran Högnäs and Arunava Mukherjea. *Probability Measures on Semigroups: Convolution Products, Random Walks and Random Matrices*, chapter Random Walks on Semigroups, pages 171–251. Springer US, 2011.

[10] Jesse Hoogland, George Wang, Matthew Farrugia-Roberts, Liam Carroll, Susan Wei, and Daniel Murfet. Loss landscape degeneracy drives stagewise development in transformers, 2024.

[11] Andrew Hryniowski and Alexander Wong. Seeing convolution through the eyes of finite transformation semigroup theory: An abstract algebraic interpretation of convolutional neural networks, 2019.

[12] David Hume. *An Enquiry Concerning Human Understanding*. Printed for A. Millar, 1748.

[13] Marcus Hutter, David Quarel, and Elliot Catt. *An Introduction to Universal Artificial Intelligence*. Chapman and Hall/CRC, April 2024.

[14] Robert M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[15] Stephen Cole Kleene. Representationof events in nerve nets and finite automata. *CE Shannon and J. McCarthy*, 1951.

[16] A. N. Kolmogorov. Three approaches to the quantitative definition of information*. *International Journal of Computer Mathematics*, 2(1–4):157–168, January 1968.

[17] Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.

[18] Kenneth Krohn and John Rhodes. Algebraic theory of machines. i. prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116(0):450–464, 1965.

[19] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 44(1.2):261–269, January 2000.

[20] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer International Publishing, 2019.

[21] Andrew Ly and Pulin Gong. Optimization on multifractal loss landscapes explains a diverse range of geometrical and dynamical properties of deep learning. *Nature Communications*, 16(1), April 2025.

[22] B. McMillan. Two inequalities implied by unique decipherability. *IEEE Transactions on Information Theory*, 2(4):115–116, December 1956.

[23] Chris Mingard, Henry Rees, Guillermo Valle-Pérez, and Ard A. Louis. Deep neural networks have an inbuilt occam's razor. *Nature Communications*, 16(1), January 2025.

[24] Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A. Louis. Neural networks are a priori biased towards boolean functions with low entropy, 2019.

[25] Daniel Murfet and Will Troiani. Programs as singularities, 2025.

[26] William of Ockham. Summa logicae (pars i, cap. 12), circa 1323.

[27] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, April 1959.

[28] Claude E Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

[29] R.J. Solomonoff. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, March 1964.

[30] R.J. Solomonoff. A formal theory of inductive inference. part ii. *Information and Control*, 7(2):224–254, June 1964.

[31] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937.

[32] Guillermo Valle-Pérez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions, 2018.

[33] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability &; Its Applications*, 16(2):264–280, January 1971.

[34] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, August 2009.

[35] Sumio Watanabe. *Mathematical theory of Bayesian statistics*. Chapman and Hall/CRC, 2018.

[36] Ingo Wegener. *The complexity of Boolean functions*. John Wiley & Sons, Inc., 1987.

[37] Susan Wei, Daniel Murfet, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that's good. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10473–10486, December 2023.